Syntax

1 syntax

參考：java grammer (oracle, chapter 18)

關鍵字(keyword)
變數(variable)、常數(constant): 命名
表達式(express): 加減乘除，邏輯運算
敘述(statement): 換行或分號。
流程控制(flow control) if,loop,case

2 html and javascript

利用<script>... </script> HTML標籤，表示javascript 區塊。利用這個標籤，瀏覽器知道這是程式碼。例如，

```
<script ...>
   JavaScript code
</script>
```

這個標籤有兩個重要屬性（attributes）:-

- **Language** – 使用何種scripting語言，內訂為javascript，但是新標準的HTML有不用的趨勢。
- **Type** – 目前建議為此種屬性，通常設定為"text/javascript".

例如，

```
<script language="javascript" type="text/javascript">
   JavaScript code
</script>
```

HTML文件由標籤組成，而這些規格的設定由於演進的關係，一直在變，因此學習HTML文件的時候會出現一樣的HTML在不同的瀏覽器不一致的情況，例如

html 的script標籤，在HTML5的定義中，只要<script> code here </script>
但是之前標準HTML還有其他語法定義，例如
<script type="text/javascript"> code here </script>

又，寫成<script type="javascript"> 也不行。

```
<script type='javascript'>
alert('dd')
</script>
```

✘上面的程式不執行。

```
<script type='text/plain'>
alert('dd')
</script>
```

✘上面的程式不執行。

因此對於初學者而言，限制在某個瀏覽器，和某個版本有助於簡化HTML的應用。

另外，利用HTML解譯器的特性，如果寫成這樣

<script type="Test">

雖然沒有符合MIME的任何類型，HTML 解譯器也不會錯誤因而終止。因此，可以用來做為註解。因為browser會跳過這個<script>區塊，因此裡面的任何程式碼也不會被執行。(但是chrome採用hmtl5，預設是是javascript 而執行。因此可以只寫<script> </script>。

也就是說如果我們這樣寫

<script type="text/plain">

可以用來不被執行，但是程式碼仍然保留。

除了語法上的改變，甚至瀏覽器對標籤的解讀也不見的一樣，例如瀏覽器再載入script區塊的時候，內定是不隱藏的，但是有的瀏覽器可以藉由對css 的定義，而讓script區塊是可見的。

例如

```
<style>
 script {
   display:block
 }
</style>
```

type="text/plain"

Multipurpose Internet Mail Extensions(MIME)類型		
超文本標記語言文本	.htm,.html	text/html

普通文本	.txt	text/plain
RTF文本	.rtf	application/rtf
GIF圖形	.gif	image/gif
JPEG圖形	.ipeg,.jpg	image/jpeg
MPEG文件	.mpg,.mpeg	video/mpeg
AVI文件	.avi	video/x-msvideo
GZIP文件	.gz	application/x-gzip
TAR文件	.tar	application/x-tar

在windows 系統中，控制台\程式集\預設程式\設定關聯(建立檔案類型或通訊協定與程式之間的關係)

note:無法用瀏覽器開啟的MIME檔案，必須另外使用其他應用程式開啟的檔案，就是application/(xxx)

2.1 JavaScript 的撰寫位置

HTML 的基本結構如下

```
<html>
    <head></head>
    <body></body>
</html>
```

程式碼可以位於

- <head> 區塊中，
- <body>區塊中。
- 也可以同時放置於上述兩個區塊。
- 外部檔案

<head>

```
<html>
   <head>
      <script>
            var a;
            a=3;
      </script>

   </head>
```

```
   <body>
   </body>

</html>
```

放在函數中比較恰當

```
<html>
   <head>
      <script type="text/javascript">
            function sayHello() {
               alert("Hello World")
      </script>

   </head>

   <body>
      <input type="button" onclick="sayHello()" value="Say Hello" />
   </body>

</html>
```

<body>

```
<html>

   <head>
   </head>

   <body>

      <script type="text/javascript">
         <!--
            document.write("Hello World")
         //-->
      </script>

      <p>This is web page body </p>

   </body>
</html>
```

同時存在於 <body> 和 <head>

也可以同時在<head> <body> 段落中使用

```
<html>
   <head>
      <script>
        var a;
```

```
        a=3;
     </script>
   </head>

   <body>
      <script>
         var b;
         b=a+4;
     </script>

   </body>
</html>
```

外部檔案

我們也可以利用<script>標籤的屬性src 將javascript寫在外部檔案，例如

```
<html>

   <head>
      <script src="filename.js" ></script>
   </head>

   <body>
      .......
   </body>
</html>
```

而外部檔案的副檔名通常是 .js。例如下面的操作

將下面的函數，放在檔案**filename.js**

filename.js

```
function sayHello() {
   alert("Hello World")
}
```

雖然，可以在這兩個部分撰寫程式碼。但是，有順序關係。可以利用debug 工具(chrome 開發人員工具)幫助瞭解。

inline code

```
<html>
   <body>
      <input type="button" onclick="javascript:alert('hello');" value="Say Hello" />
      <input type="button" onclick="alert('hello');" value="Say Hello(HTML5)" />


   </body>
```

```
</html>
```

2.2 Your First JavaScript Script

程式碼只有一行

但是用<!-- //--> 把程式碼圈起來。用來讓不支援javascript的瀏覽器把這段程式碼視為註解。這只是一個技巧，在HTML中，<!-- --> 表示註解，但是後面的-->前面又加上//，這是要讓javascript 把「-->」一整行都忽略。

【例】

```
<html>
   <body>
      <script>
         <!--
            document.write("Hello World!")
         //-->
      </script>
   </body>
</html>
```

結果

```
Hello World!
```

note:

document 指的是瀏覽器當時active的一頁（不是原本檔案的內容，因為內容可能是由javascript動態產生）。以物件導向的觀點來看，document 是一個物件， write 是 document 物件的一個方法， 利用一點 (.)，來連結。

完整的寫法

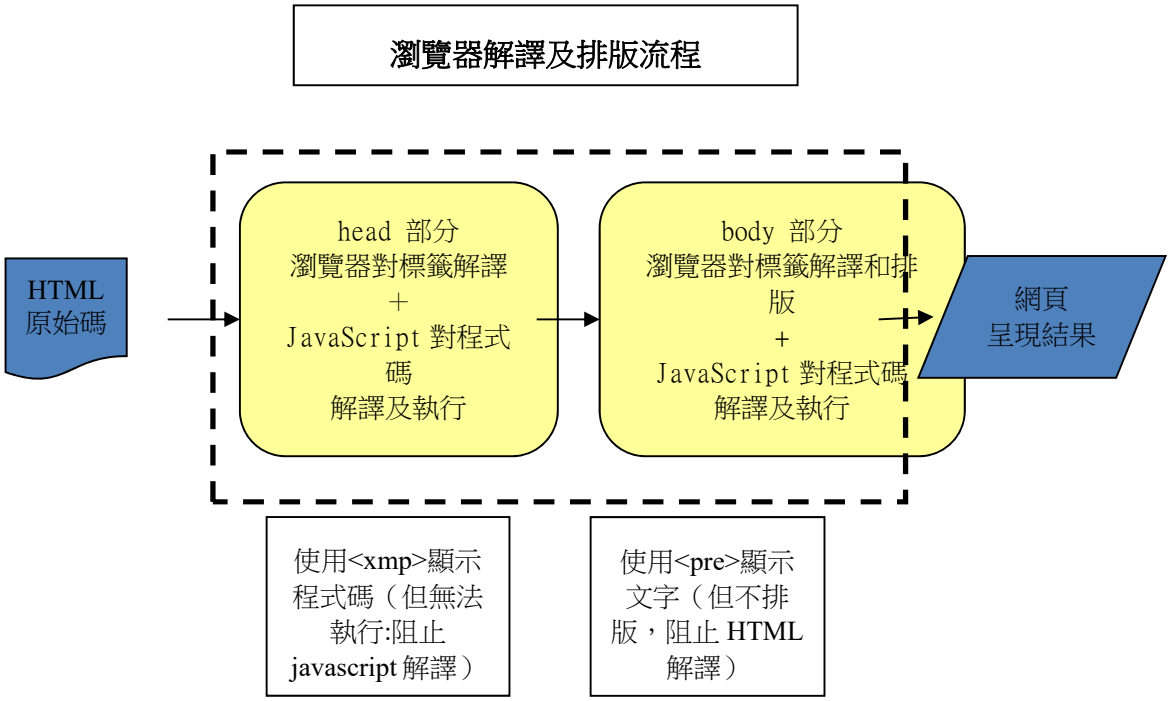
```
<html>
<head>
<title> 範例</title>
</head>
<body>

<script>
<!--
document.write("Hello, World. ");
//-->
</script>

</body>
</html>
```

2.3 小結

在HTML中適合撰寫 JavaScript 的位置有：

- <head> ，優點是可以保證在載入 <body> 內容之前已經完全載入 <head> 的
- <body>，
- </html> 一般免費網頁供應商在此加入彈窗 (pop-up window) 。

但是位置的差異，和瀏覽器的運作過程有關：



【例】討論下面的程式碼，測試程式載入的順序

需要知道什麼是函數。

```
<html>
   <head>
      <script>
            function sayHello1() {
               alert("Hello World:in head")
            }
               sayHello2(); //這裡會出錯
      </script>
   </head>

   <body>
```

```
      <input type="button" onclick="sayHello1()" value="呼叫sayHello11(in head)" />
    <br>
    <input type="button" onclick="sayHello2()" value="呼叫sayHello12(in body)" />
      <script>
            function sayHello2() {
               alert("Hello World:in body")
            }
            sayHello1();
      </script>
   </body>
</html>
```

2.4 Whitespace and Line Breaks

在javascript中，空白（包括tab）和分行都會被忽略。因此，利用這些東西可以讓程式碼的編寫，具有可讀性。

2.5 分號

類似C，C++，Java，基本上每個javascript敘述(statements)都以分號分隔，但是如果能在一行內寫完，可以不用分號。例如

```
<script
   <!--
      var1 = 10
      var2 = 20
   //-->
</script>
```

但是，如果要寫在同一行，那麼就一定要分號，如下

```
<script language="javascript" type="text/javascript">
   <!--
      var1 = 10; var2 = 20;
   //-->
</script>
```

Note – It is a good programming practice to use semicolons.

2.6 大小寫分別

例如，關鍵字 **Time** 和 **TIME** 在javascript中有不同的意義。

2.7 註解

JavaScript 的註解，承襲C 和 C++的註解方式：

- 單行註解在符號 // 和換行之間的所有文字，都忽略。其他語言例如VB的單行註解'，matlab 則是 %
- /* 和 */ 也都忽略，可以是多行。

Example

```
<script>
   <!--

      // 類似C++ 的註解；只針對一行，

      /*
       多行註解
      */

   //-->
</script>
```

關於上例子中，HTML註解：<!-- -->

- JavaScript 也認得 HTML的註解的開始符號<!--，同時把之視為類似//的用法。
- 但是HTML 註解的結束符號--> 是不認得的，因此要加上// (但是在chrome 中，可以)

3 宣告

一般程式語言的宣告，可以分成變數宣告和函數宣告。

3.1 JavaScript Datatypes

在程式語言中，通常一個重要準備工作就是，這個語言支援了哪些資料型態：JavaScript 只有三種基本資料型態：

- **Numbers,** eg. 123, 120.50 etc.
- **Strings** of text e.g. "This text string" etc.
- **Boolean** e.g. true or false.

在JavaScript中，整數和浮點數字沒有區別，在JavaScript中，所有的數字都以浮點的方式存取 (定義在IEEE 754 標準中的 64-bit floating-point format)。

此外，JavaScript 還有一些特殊數值如下：

- NaN (Not a Number，非數字): 用不正確的資料 (例如字串或未定義的變數) 來執行數學運算時，或是執行無意義的數學運算 (例如 0/0)，就會產生這個數值
- 正的無限大 (Infinity): 當正數大到無法顯示在 JavaScript 中時，就會使用這個數值
- 負的無限大 (-Infinity): 當負數大到無法顯示在 JavaScript 中時，就會使用這個數值

如果上述的資料型態可以視為是「型態」的基本元素，則陣列(array)和物件（Object）就是組合式的型態。

- array
- object (function)

```
var  a,b;
a=[]; //空陣列
b = {};//  空物件
```

另外，與資料型態有關的兩個關鍵字是 **null** 和 **undefined** 。

例如

```
var a;
console.log(a)  //結果 undefined
```

3.2 變數宣告

所有的程式語言都會有變數(variables)。變數只是對記憶體位置的一個命名，可以用來存放前面提及的各種型態的資料。而相關的關鍵字就是: **var** ，如下：

```
<script type="text/javascript">
   <!--
      var money;
      var name;
   //-->
</script>
```

也可以定義在同一行：

```
<script type="text/javascript">
   <!--
      var money, name;
   //-->
</script>
```

宣告變數的時候，也可以同時指派資料值，下面的範例有兩種指派位置：

```
<script type="text/javascript">
   <!--
      var name = "Wugo";
      var money;
      money = 1200.50;
   //-->
</script>
```

Note – 通常宣告一個變數以後，在整個程式的執行過程中不會再重複宣告一次。

JavaScript 是一個untyped語言 （不是 no type）意思是說，在javaScript執行過程中，一個變數可以指派任意資料型態(data type)。

```
<script type="text/javascript">

      var x ;
      x= "Wugo";
      x= 1200.50;
```

</script>

java:char,byte,boolean,int,long,float,double

int x=3; //32bit
long x=3;//64bit
float x=3.2;//32bit
double x =3.2; 64bit

vb:long,integer,double,string,char,boolean

dim x as long
x=3
dim y as integer
y=3
dim z as double
沒有float

3.3 小結論 : small introduction to JavaScript's type system

在JavaScript 中的值，有兩種基本類別： primitives 和objects.
Strings、 numbers 、booleans 、null 、undefined 都是primitives，而物件是那些有properties 的任何東西。甚至陣列和函數都是一般的物件，因此都可以有屬性欄位。
在JavaScript中，primitive 看起來好像事物件的原因是autoboxing，但是不論如何，primitives 就是不會有屬性。

例如，

```
var a = 'quux';
a.foo = 'bar';
document.writeln(a.foo);
```

這裡的結果是"undefined"，因為: a 被指派了一個 primitive 值，當被指派了一個欄位foo的時候，自動升格為物件。但是這個物件在writeln(a.foo)的時候，並不是上一行a.foo=bar 的物件，而是一個全

新的物件。因此，foo的值遺失。

可以想成這樣：

```
var a = 'quux';
new String(a).foo = 'bar';
document.writeln(new String(a).foo); // 一個全新的物件在這裡又建立。
```

3.4 JavaScript 的變數範圍（Variable Scope）

JavaScript 只有兩個變數範圍：

- **Global Variables** – 在整個程式範圍都可以存取的變數。
- **Local Variables** – 指的是只有在副程式範圍可見的變數，例如函數參數。

變數衝突，如果局部變數和總體變數衝突，總是局部變數優先。例如：

```
<html>
   <body onload = checkscope();>
      <script type = "text/javascript">
            var myVar = "global"; // Declare a global variable
            function checkscope( ) {
               var myVar = "local";  // Declare a local variable
               document.write(myVar);
            }
      </script>
   </body>
</html>
```

結果：

```
local
```

測試下面的程式

```
<html>
   <body >
      <script type = "text/javascript">

            var myVar = "global"; // Declare a global variable
            function checkscope( ) {

               var myVar = "local";  // Declare a local variable
               document.write(myVar);

            }
          checkscope();
          document.write(myVar);

      </script>
```

```
   </body>
</html>
```

【例】

```
<script type="text/javascript">
  var myVar = "global"; // Declare a global variable
  function varTest() {
    var x = 1;
    if (true) {
      var x = 2; // same variable!
      console.log(x); // 2
    }
    console.log(x); // 2
  }

  function checkscope() {

    var myVar = "local";
    document.write(myVar); // 輸出：local

  }
  checkscope();
  varTest();
  document.write(myVar); // 輸出：global
</script>
```

上面執行的結果是
2
2
local
global

結論是var 的活動範圍不以區塊分而是以函數範圍來區分。如果要以區塊來分，可以使用關鍵字"let"。

3.5 JavaScript 命名規則

While naming your variables in JavaScript, keep the following rules in mind.

- 不可使用關鍵字。
- 一定是英文字母開頭（包括_）。.
- 大小寫不一樣。

例如，

```
var _x,_x3, _3, $x;
```

錯誤的方式

```
var red ball, 3b;
```

不能有空白，不可以數字開頭。

3.6 JavaScript Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

基本輸入，輸出

```
<html>

   <head>
   </head>

   <body>

      <script type="text/javascript">
         <!--
            document.write("Hello World")
         //-->
      </script>

      <p>This is web page body </p>

   </body>
</html>
```

3.7 undefined

「undefined」是Javascript定義的特殊資料型態，可以判斷兩種情況

- 未宣告(記憶體中沒有這個變數的位置)，且未初始化。
- 已宣告，且未初始化。

3.8 typeof

如果變數沒有宣告，直接利用函數typeof()判斷型態，則傳回值為"undefined"(「字串」)，此時a沒有「值」

例如，

```
<script>
  typeof(a)
</script>
```

結果

"undefined"

但是，如果已經經過宣告(var a;)，但未初始化(尚未址派值)，則javascript自動給值a=undefined（非字串），但是typeof(a) 仍然給"undefined"(字串)；

```
<script>
  var a; //此時a 的內容或值為undefined(非字串)
  typeof(a)
</script>
```

結果

"undefined"

```
<script>
// notDeclare 變數未經過宣告，所以無值
// 此時 typeof(notDeclared) 會傳回 "undefined" 字串
if (typeof(a) == "undefined")
  document.write("a 未被宣告<br>");
document.write("typeof(a)="+typeof(a)+"<br>");

// declare 變數只經過宣告，但尚未初始化，它的值是 undefined（非字串！）
// 此時 typeof(declared) 仍會傳回 "undefined" 字串
var declared;
if (declared==undefined)
  document.write("declared 已宣告但未被啟始<br>");
document.write("declared="+declared+"<br>");
document.write("typeof(declared)="+typeof(declared)+"<br>");
</script>
```

3.9 null

對照C/C＋＋語言來看，null的意思是空指標。指標是一個抽象名詞，直觀理解就是記憶體的一個位置索引。null意思為不指向任何位置。

hint
typeof(null) :object
null==false : false
null == 0 : fasle

!null : true

```
<pre>
<script>
x=null; //不是Null 也不是NULL;

document.writeln("x 的型態："+typeof(x)); //object ,空指標
//check 1
document.write("null 是否等於false，即null==false的執行結果為：");
document.writeln(null==false);//  null=0=false
//check 2
document.write("x 是否等於false，即x==false的執行結果為：");
document.writeln(x==false);//  null=0=false
//check 3
if (x)
  document.writeln("在條件判斷if(x) 中，x被轉換為true");
else
  document.writeln("在條件判斷if(x) 中，x被轉換為false");
</script>
</pre>
```

結果：

//check 1

x 的型態：object

//check 2

null 是否等於false，即null==false的執行結果為：false

//check 3

x 是否等於false，即x==false的執行結果為：false

在條件判斷if 中，x被轉換為false

<註>

在 js 裡反而比較少直接返回 null 的

以下的例子會返回 null

```
object.getPrototypeOf(Object.prototype)
```

</註>

4　輸入/輸出

輸出

```
      <script>
       var a;
       a=3;
     console.log(a);
     document.write(a);
               console.write(b);
```

```
    alert(a);

      a="3";
    console.log(a);
    document.write(a);
    alert(a);

    </script>
```

輸入

```
prompt()
confirm()
```

confirm ("提示字串") 會彈出視窗，視窗有兩個按鈕，「確定」和「取消」，如果使用者按確定，則傳回true ，否則 false。例如，

a= confirm ("繼續？")

a的值不是true,就是false。

【練習】

從使用者拿到兩個數字

然後要求 first個數字 % (module) 地2個數字的於因子 ，例如：

5 % 3 ->2

8 % 7--->1

把結果顯示在螢幕上。

```
<html>
   <head>
      <script>
       var a,b,c;
       a=prompt("first number");

       b=prompt("second number");
     a=parseFloat(a);
     b=parseFloat(b);
     c=a%b;

     alert("first(="+a +") % " +" second("+b+") = "+c);

     </script>
   </head>
```

```
   <body>

      <script>
         var b;
         b=a+4;
      </script>

   </body>
</html>
```

5 轉換函數

字串和數字的互相轉換

parseFloat()
parseInt()

5.1 Converting Decimal to Binary

Example

```
function dec2bin(dec){
      return (dec >>> 0).toString(2);
  }
```

Try it Yourself »

> 測試

```
      <script language="javascript" type="text/javascript">
            document.write((5).toString(8));  //✅
            document.write((5>>>0).toString(8)); //✅
            document.write(5.toString(8));  //❌
      </script>
```

5.2 Converting Binary to Decimal

Example

```
function bin2dec(bin){
      return parseInt(bin, 2).toString(10);
  }
```